IN THE SPECIFICATION

Please replace the passage from page 11, line 208 through Page 12, line 248 with the following:

Fig. 3 through 6 illustrate an embodiment of an authentication process using OPIE. The user first issues a login request, e.g. by typing the following special URL:

"https://" followed by the string "absent.research.att.com/login=user"

ABSENT 210 forwards the request to PUSHWEB 220 which recognizes "login" as a special command code to begin the authentication procedure. PUSHWEB 220 negotiates a secure connection (e.g. using SSL) with the terminal 110, while ABSENT 210 blindly forwards data packets between them. ABSENT 210 essentially acts as a wire. This is similar to SSL tunneling; however, here packets are simply being forwarded without using the SSL tunneling protocol. This advantageously avoids having the user access the proxy settings of a browser on the terminal. Once the secure connection is established, PUSHWEB 220 looks up the user in a database. If the user is registered, PUSHWEB 220 submits a request to the authentication server 130 which generates an authentication challenge. PUSHWEB 220 then constructs an HTML page with a form for the user to enter a response to the authentication challenge and send the document over the secure connection to the terminal 110. Fig. 3 shows an example of a challenge page using OPIE. An OPIE challenge is of the form:

otp-md5 494 st0993 ext

where `otp-md5` indicates that MD5 is the hash function, `494` is the number of times to iterate the function, and `st0993` is the seed for the generator. A list of one-time passwords (e.g. see Fig. 5) can be printed on paper in advance or computed using a calculator. Fig. 4 illustrates a Windows-based OPIE calculator program. The user enters the seed, the number of times to iterate and the secret passphrase into the OPIE calculator which then computes and provides the one-time password. The user types the one-time password into the HTML form in Fig. 3, chooses a starting internal server document, and clicks the submit button. The form constructs a response from this information using a special code, “OTP_response” that will be recognized by PUSHWEB 220. For example, the response can be in the form:

”https://” followed by the string ”absent.research.att.com/OTP_response=user/?response=RESPONSE&startpage=URL” PUSHWEB 220 sends the response to the authentication server 130. If the authentication succeeds, then an entry is created in a user table and the page requested in the challenge form is returned to the user, e.g. Fig. 6. For security reasons, one of the fields in the user table entry can be an expiration time, which can be set to a value such as 20 minutes. After 20 minutes of usage has elapsed, the user is presented with another challenge page.

Please replace the passage from page 13, line 266 through Page 14, line 279 with the following:

Fig. 6 sets forth an example of an internal server HTML document returned by PUSHWEB 220 to the user of the terminal 110. Web pages such as the one shown in Fig. 6 may contain hyperlinks to other pages behind the firewall. Resource addresses in links (and otherwise) are conventionally expressed as Uniform Resource Locators (URLs), the format of which is described in Berners-Lee, T., et al., “Uniform Resource Locators,” RFC 1738, Network Working Group, 1994, which is incorporated herein by reference. If the links are not changed by the system, then future requests will not be able to access internal pages without re-authentication. For example, a link in an HTML document may be of the form:

[[<]] a href=”http://myhost.research.att.com/proprietary.html”>Business Plan</a[[>]] enclosed by < >.

When the user clicks on “Business Plan”, the terminal will attempt to connect directly to machine “myhost” and, of course, the firewall will not allow this.

Please replace the passage from page 14, line 290 *through Page 15, line* 314*with the following:*

An advantageous format for rewriting a URL could include the following information: (a) the proxy URL (e.g. https://absent.research.att.com) which points user terminals to the relevant proxy address and port; (b) a command, e.g., “cmd=user” where “cmd” indicates the action to be taken by PUSHWEB (“geturl”, “login”, “logout”, “OTP_Response”) and “user” is an account identification for the user); (c) security data, further described in detail below; and (d) the original URL that was contained in the page, converted to an absolute URL where necessary. As an example, the URL

”http://” followed by the string “www.research.att.com/projects”

can be rewritten as

“https://” followed by the string
“absent.research.att.com/geturl=user/2b5db86c1f6e/http://www.research.att.com/projects/”
Accordingly, PUSHWEB 220 checks whether content it is retrieving from internal server 120 might contain hyperlinks (e.g. by checking if “Content-type” of the response from the server is “text/html”). If so, the document is parsed to identify all links on the page (if not, then the response from the server can remain unedited.). Each link containing a relative URL is converted to one containing an absolute URL (e.g. by adding the “http://” protocol, adding the complete server name such as “music.research.att.com”, and/or appending the proper path information such as “/dir/foo.html” rather than “../foo.html”). URLs that are outside of the trusted domain are not changed further. However, URLs that are behind the firewall are reconstructed in the above format to prepend the relevant information needed for processing.

Please replace the passage from page 17, line 342 through Page 17, line 363 with the following:

When a user submits information to the server in a form using a GET method, a URL containing the names and values of the input boxes is passed from the terminal to the server. This URL cannot be MACed in advance by PUSHWEB because there is no way to know what values the user will enter. For example, say that a CGI script is referenced by the following URL (where “absent” represents “absent.research.att.com”):

”http://” followed by the string
“absent/geturl=alice/32a5d386cf6e/http://www.research.att.com/~alice/cgi-bin/reg.cgi”

The URL sent to the server when the user submits could be "http://" followed by the string "absent/geturl=alice/32a5d386cf6e/http://www.research.att.com/~alice/cgi-bin/reg.cgi&name=bob".

The MAC is correct for the first 50 bytes of the original URL, but it does not include "&name=bob". Therefore, it is advantageous to include, in hex, the original URL that references the CGI script. In the above example, the length would be 50, which is hex 32. The most significant 40 bits of the actual MAC are 0xa5d386cf6e. The effect is that authenticated users can execute CGI scripts as long as their keys are valid. A CGI script that is MACed by an expired key cannot be invoked. A potential danger is that an attacker who can surmise that a particular request is a CGI form can replay the message to cause the script to execute again with the same input data as before. Fortunately, SSL protects against reply attacks, so the present system is not vulnerable to this.